课程回顾

- 动态规划问题：0-1背包
- 贪心算法
  - 贪心算法可求解问题的特性
  - 活动选择问题：动态规划算法 vs. 贪心算法

贪心算法内容

- 活动选择问题
- 贪心算法原理
- 分数背包问题
- Huffman编码
- 拟阵
- 其他应用

贪心算法原理

- 贪心算法是通过做一系列选择来获得最优解，在每个决策点，都选择在当时看来最佳的选择，这种启发式策略并不保证总能产生最优解
- 前述贪心算法的步骤
  1. 确定问题的最优子结构
  2. 给出递归解
  3. 证明在递归的每一步，有一个最优的选择是贪心的选择，因此做出这种选择是安全的
  4. 证明除了贪心选择导出的子问题外，其余子问题都是空集合
  5. 根据贪心策略写出递归算法
  6. 将递归算法转换为迭代算法

贪心算法原理 (续)

- 上述步骤是以动态规划为基础的。实际上可改进它，重点关注贪心选择
- 例：活动选择问题可改进为：
  - 首先定义子问题S_{ij}，i, j均可变
  - 如果我们总是做贪心选择，则子问题变为：$S_i=\{a_k \in S: f_i \le s_k\}$（只需考虑$S_{ij}$中$i$或$j$一端变化情况）
  - 证明一个贪心的选择（即选S_i中第一个完成的活动a_m），和剩余子问题S_m（与a_m兼容）的最优解结合，能产生S_i的最优解

贪心算法原理 (续)

- 贪心算法的一般设计步骤
  - 将优化问题分解为做出一种选择及留下一个待解的子问题
  - 证明对于原问题总是存在一个最优解会做出贪心选择，从而保证贪心选择是安全的
  - 验证当做出贪心选择之后，它和剩余的一个子问题的最优解组合在一起，构成了原问题的最优解
- 没有一般的方法说明贪心算法是否会解决一个特殊的最优问题，但是有两个要素有助于使用贪心算法：
  - 贪心选择性质
  - 最优子结构

贪心算法原理 (续)

■贪心选择性质

- 贪心选择性质：一个全局最优解能够通过局部最优（贪心）选择达到
- 贪心法总是在当前步骤上选择最优决策，然后解由此产生的子问题
- 贪心选择只依赖了目前所做的选择，但不依赖于将来的选择及子问题的解
- 自顶向下，每做一次贪心选择，就将子问题变得更小
- 贪心算法一般总存在相应的动态规划解，但贪心法的效率更高，原因：
  - 对输入做预处理
  - 使用合适的数据结构（如优先队列）

贪心算法原理 (续)

- 最优子结构
  - 和动态规划一样，该性质也是贪心算法的关键要素
    例：活动选择问题的动态规划解中：
    S_{ij}的最优解A_{ij}：若$a_k \in A_{ij}$，则A_{ij}包含S_{ik}和S_{kj}的最优解
  - 对贪心算法更直接

贪心算法原理 (续)

- 贪心法与动态规划的比较
  - 相同点：两种方法都利用了最优子结构特征
  - 易错误处：
    - 当贪心算法满足全局最优时，可能我们试图使用动态规划求解，但前者更有效
    - 当事实上只有动态规划才能求解时，错误地使用了贪心算法
- 为了说明两种技术的细微区别，我们研究一个经典最优化问题的两个变形：
  - 0-1背包问题（动态规划章节已讲述；教材p243）
  - 分数背包问题（每个物品可以拿取部分）

贪心算法原理 (续)

■两个背包问题都具有最优子结构!

➢0-1背包问题：

原问题的最优解：重量至多为W的最有价值的物品（取自n件物品）的装包方案，每个物品只能选择取或不取

子问题：求解背包容量为j，可选物品编号为1, 2, ..., i时的0-1背包问题

子问题的最优解：从原问题的最优解中去掉某物品j，则包中剩余物品应是除j外、取自原n-1件物品中最有价值、且总重不超过W-w_j的若干物品

原问题的最优解也要求子问题的解最优

贪心算法原理 (续)

■两个背包问题都具有最优子结构!

➢分数背包问题：

原问题的最优解：重量至多为W的最有价值的物品（取自n件物品）的装包方案，每个物品可部分拿取

子问题：求解背包容量为j，可选物品编号为$1, 2, ..., i$时的分数背包问题

子问题的最优解：从背包中去掉包含物品j（可能是物品j的一部分），设其重量为w，则包中剩余物品应是除该部分之外、取自原n-1件物品以及物品j的w_j-w部分中最有价值、且总重至多为W-w的若干物品

原问题的最优解也要求子问题的解最优

贪心算法原理 (续)

- 求解0-1背包问题最优解只能使用动态规划
- 求解分数背包问题最优解可使用贪心算法：
  - 将物品按照单位重量价值（v_i/w_i）排序
  - 每次取单位重量价值最大物品装包，直至背包装满或所有物品都考虑过为止

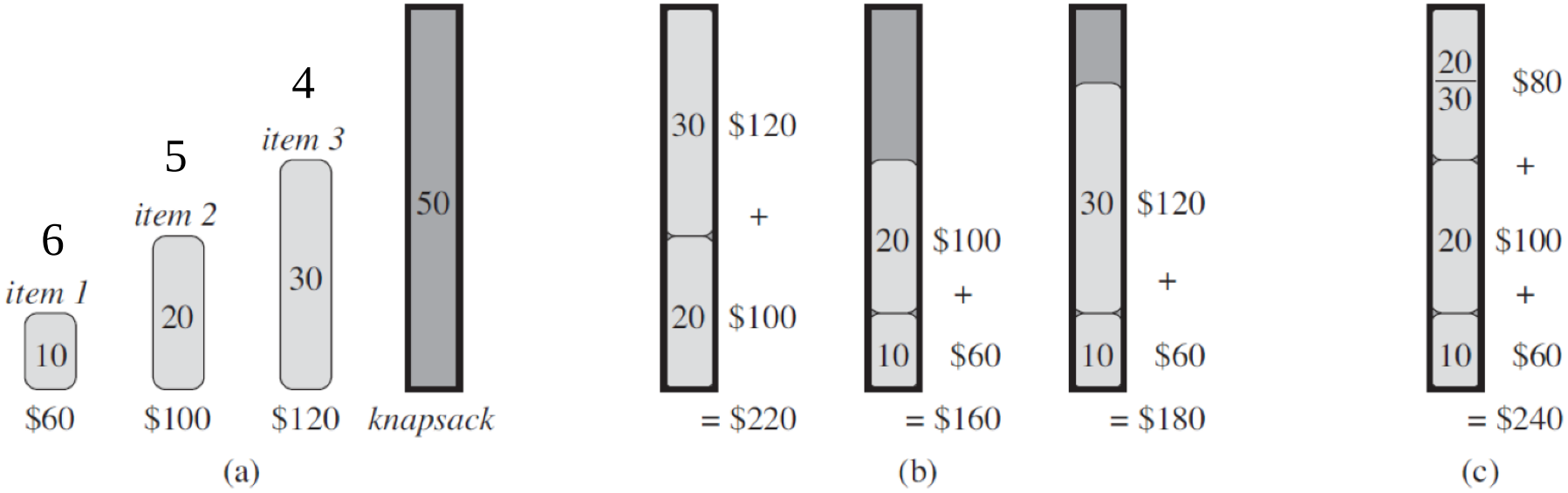
该朴素贪心算法若用于0-1背包问题，所得的解任意差！

例：

	物品1	物品2
重量w_i	1	100
价值v_i	2	199
单位重量价值v_i/w_i	2	1.99
背包容量W	100	

只装入物品1？

贪心算法原理 (续)


6
item 1
10
$60
5
item 2
20
$100
4
item 3
30
$120
50
knapsack
(a)
30 $120
+
20 $100
= $220
20 $100
+
10 $60
= $160
30 $120
+
10 $60
= $180
(b)
20/30 $80
+
20 $100
+
10 $60
= $240
(c)

贪心算法内容

- 活动选择问题
- 贪心算法原理
- 分数背包问题
- Huffman编码
- 拟阵
- 其他应用

贪心算法理论

- 该理论可用来确定贪心算法生成最优解的情形
- 用到了一种组合结构：Matroid（拟阵）
  - 该结构是Whitney于1935年在研究矩阵中线性无关结构时抽象出来的，由Korte于80年代初将该理论发展为贪心算法理论
- 该理论覆盖了许多贪心算法的应用实例（Kruskal、匈牙利算法等），但并未覆盖所有情况（如活动选择问题、Huffman编码等），仍在发展中

贪心算法理论 (续)

- 一个拟阵是满足下列条件的有序对$M = (S, \mathcal{I})$：
  - S是一个有限集
  - $\mathcal{I}$是S的子集的一个非空族，这些子集称为S的独立子集，使得若$B \in \mathcal{I}$且$A \subseteq B$，则$A \in \mathcal{I}$。若$\mathcal{I}$满足此性质，则称之为遗传的，注意空集必然是$\mathcal{I}$的成员，即$\emptyset \in \mathcal{I}$
  - 若$A \in \mathcal{I}$、$B \in \mathcal{I}$且$|A| < |B|$，那么存在某个元素$x \in B - A$，使得$A \cup \{x\} \in \mathcal{I}$，则称$M$满足交换性质

贪心算法理论 (续)

■拟阵 $M = (S, \mathcal{I})$：

- S有穷非空
- 集合族$\mathcal{I}$满足遗传性，即某子集是独立子集，则该独立子集的子集也是独立子集
- M满足交换性质，即可扩展

贪心算法理论 (续)

■例：$M = (S, \mathcal{I})$

➢矩阵拟阵：S的元素是矩阵的行，若行的子集线性无关则该子集独立

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 6 & 8 & 10 \\ 1 & 3 & 5 & 7 & 9 \end{bmatrix}$$

$$S = \{[1,2,3,4,5], [2,4,6,8,10], [1,3,5,7,9]\}$$

$$\mathcal{I} = \{\emptyset, \{[1,2,3,4,5]\}, \{[2,4,6,8,10]\}, \{[1,3,5,7,9]\}, \{[1,2,3,4,5], [1,3,5,7,9]\}, \{[2,4,6,8,10], [1,3,5,7,9]\}\}$$

贪心算法理论 (续)

■例：$M = (S, \mathcal{I})$

➢无向图$G = (V, E)$的图拟阵：$M_G = (S_G, \mathcal{I}_G)$

- S_G是G的边集，即$S_G = E$
- 设A是E的子集，$A \in \mathcal{I}_G$ 当且仅当 A无回路
  即：边集A独立 当且仅当 子图$G_A = (V, A)$形成森林

M_G与最小生成树问题MST紧密相关

贪心算法理论 (续)

■最大独立集

➢独立子集的扩展：在拟阵$M = (S, \mathcal{I})$中，若$A \in \mathcal{I}$且$x \in S - A$，若$A \cup \{x\} \in \mathcal{I}$，则元素$x$称为$A$的一个扩展，即元素$x$扩充到独立子集$A$后仍保持独立性

- 例：在图拟阵M_G中，若A是一个独立子集，则e是A的扩展是指加入e后仍不产生回路

➢拟阵的最大独立子集

- 若A是拟阵M的独立子集，且无法进行任何扩展，则A称为M的最大独立子集，即在M中没有更大的独立子集能包含A

贪心算法理论 (续)

■**定理16.6** 拟阵中所有最大独立子集都具有相同大小

证明（反证法）：假设拟阵M存在一个最大独立子集A和另一个更大的独立子集B
由交换性质：对于某个$x \in B - A$，有$A \cup \{x\} \in \mathcal{I}$，即$A$可扩展，与$A$是最大独立子集矛盾

贪心算法理论 (续)

■加权拟阵：若对$\forall x \in S$，为x指派一个正的权值$w(x)$，则称$M = (S, \mathcal{I})$是加权拟阵。S的子集（独立子集）的权重函数可定义为：

$$w(A) = \sum_{x \in A} w(x), \quad \forall A \subseteq S$$

➢例：若令$w(e)$表示图拟阵M_G中边e的权重（例如边的长度等），那么$w(A)$为A中所有边的权重之和

贪心算法理论 (续)

- 可用贪心算法求出最优解的很多问题（但不是全部）可形式化为在一个加权拟阵中寻找最大权重的独立子集问题。即：给定加权拟阵 $M = (S, \mathcal{I})$，希望寻找独立集 $A \in \mathcal{I}$，使 $w(A)$ 最大
- 拟阵的最优子集：拟阵中权值最大的独立子集
  最优子集一定是拟阵中的一个最大独立子集，反之不然
- 加权拟阵的贪心算法
  - 适用于任何加权拟阵求最优子集 A
  - 贪心之处：尽可能选权值最大的元素扩充到 A

贪心算法理论 (续)

```
GREEDY (M, w)
1   A ← ∅
2   按权重w单调递减对M.S中元素排序
3   for each x∈M.S, 按w单调递减次序取出x do
4       if A∪{x} ∈ M.I        //独立性检验
5           A ← A∪{x}         //扩展x未破坏独立性
6   return A
```

■时间复杂度：$O(n\lg n + nf(n))$

- 排序：$O(n\lg n)$
- 检查$A \cup \{x\}$独立性：$O(f(n))$

定理16.11说明了加权拟阵上贪心算法的正确性

贪心算法内容

- 活动选择问题
- 贪心算法原理
- 分数背包问题
- Huffman编码
- 拟阵
- 其他应用

其他应用——最优装载

- 问题：有一批集装箱要装上一艘载重量为c的轮船。其中集装箱i的重量为w_i。最优装载问题要求确定在装载体积不受限制的情况下，将尽可能多的集装箱装上轮船
- 最优装载问题可用贪心算法求解。采用重量最轻者先装的贪心选择策略，可产生最优装载问题的最优解
- 时间复杂度：$O(n\lg n)$

其他应用——单源最短路径

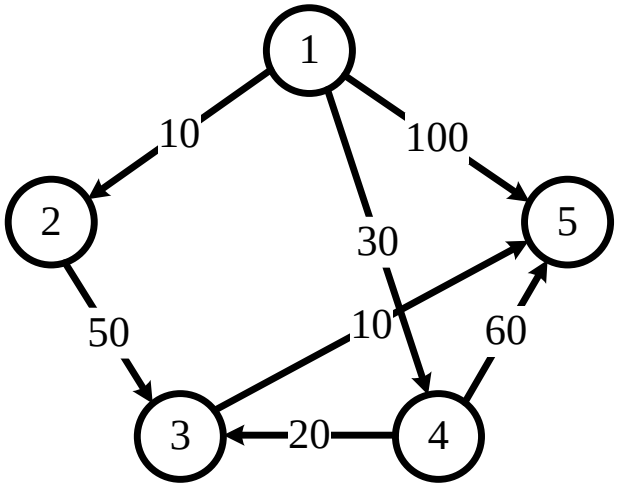
- 给定带权有向图$G = (V, E)$，其中每条边的权是非负实数
- 给定V中的一个顶点，称为源
- 单源最短路径问题：计算从源到所有其它各顶点的最短路径长度，这里路径长度是指路上各边权之和
- Dijkstra算法是解单源最短路径问题的贪心算法
  - 基本思想：设置顶点集合S并不断地作贪心选择来扩充这个集合。一个顶点属于集合S当且仅当从源到该顶点的最短路径长度已知

其他应用——单源最短路径 (续)

- 初始时，S中仅含有源
- 设u是图G的某一个顶点，把从源到u且中间只经过S中顶点的路称为从源到u的特殊路径，并用数组dist记录当前每个顶点所对应的最短特殊路径长度
- Dijkstra算法每次从V-S中取出具有最短特殊路长度的顶点u，将u添加到S中，同时对数组dist作必要的修改
- 一旦S包含了所有V中顶点，dist就记录了从源到所有其它顶点之间的最短路径长度

其他应用——单源最短路径 (续)

■Dijkstra算法实例



迭代	S	u	dist[2]	dist[3]	dist[4]	dist[5]
初始	{1}	–	10	maxint	30	100
1	{1, 2}	2	10	60	30	100
2	{1, 2, 4}	4	10	50	30	90
3	{1, 2, 4, 3}	3	10	50	30	60
4	{1, 2, 4, 3, 5}	5	10	50	30	60

其他应用——单源最短路径 (续)

- 算法的正确性
  - 贪心选择性质
  - 最优子结构性质
- 算法时间复杂度$O(|V|^2)$

其他应用——最小生成树

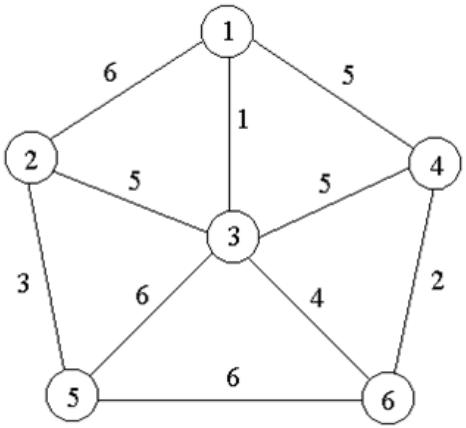
- 问题：设$G = (V, E)$是无向连通带权图，E中每条边(v, w)的权为$c(v, w)$。如果G的子图G'是一棵包含G的所有顶点的树，则称G'为G的生成树，生成树上各边权的总和称为该生成树的耗费。在G的所有生成树中，耗费最小的生成树称为G的最小生成树
- 最小生成树性质：设$G = (V, E)$是连通带权图，U是V的真子集。如果$(u, v) \in E$，且$u \in U$，$v \in V\text{-}U$，且在所有这样的边中，(u, v)的权$c(u, v)$最小，那么一定存在G的一棵最小生成树，它以(u, v)为其中一条边

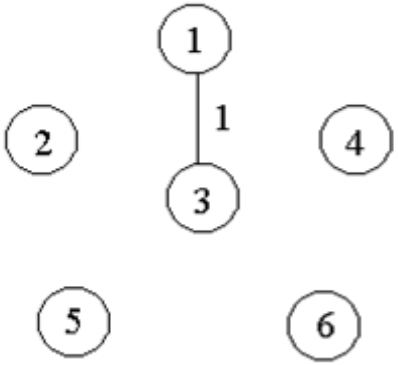
其他应用——最小生成树 (续)

■Prim算法

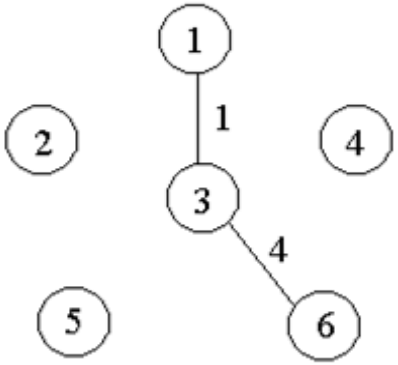
- 设$G = (V, E)$是连通带权图，$V = \{1, 2, \ldots, n\}$
- 构造G的最小生成树的Prim算法的基本思想是：首先置$S = \{1\}$，然后，只要S是V的真子集，就作如下的贪心选择：选取满足条件$i \in S$，$j \in V\text{-}S$，且$c(i, j)$最小的边，将顶点j添加到S中。这个过程一直进行到$S = V$时为止
- 在这个过程中选取到的所有边恰好构成G的一棵最小生成树
- 时间复杂度：$O(|V|^2)$

其他应用——最小生成树 (续)

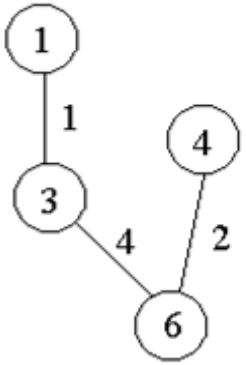




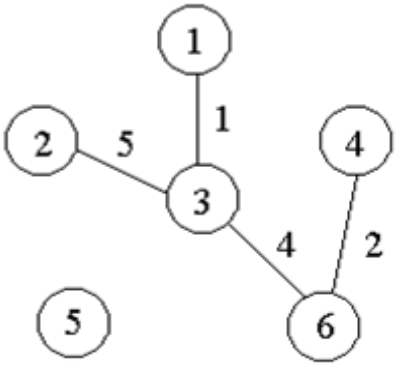
(a)



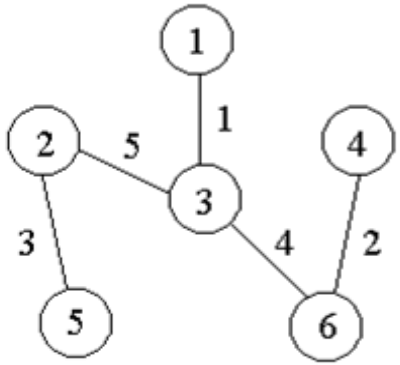
(b)



(c)



(d)



(e)

其他应用——最小生成树 (续)

■Kruskal算法

- 首先将G的n个顶点看成n个孤立的连通分支
- 将所有的边按权从小到大排序
- 按排序后顺序检查每一条边(v, w)，若v和w分别是当前2个不同的连通分支中的顶点时，则添加该边；直到只剩下一个连通分支时为止
- 时间复杂度：$O(|E|\lg|E|)$

回顾贪心算法理论

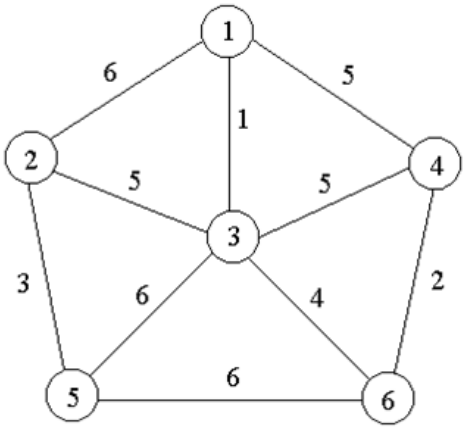
```
GREEDY (M, w)
1   A ← ∅
2   按权重w单调递减对M.S中元素排序
3   for each x∈M.S, 按w单调递减次序取出x do
4       if A∪{x} ∈ M.I        //独立性检验
5           A ← A∪{x}         //扩展x未破坏独立性
6   return A
```

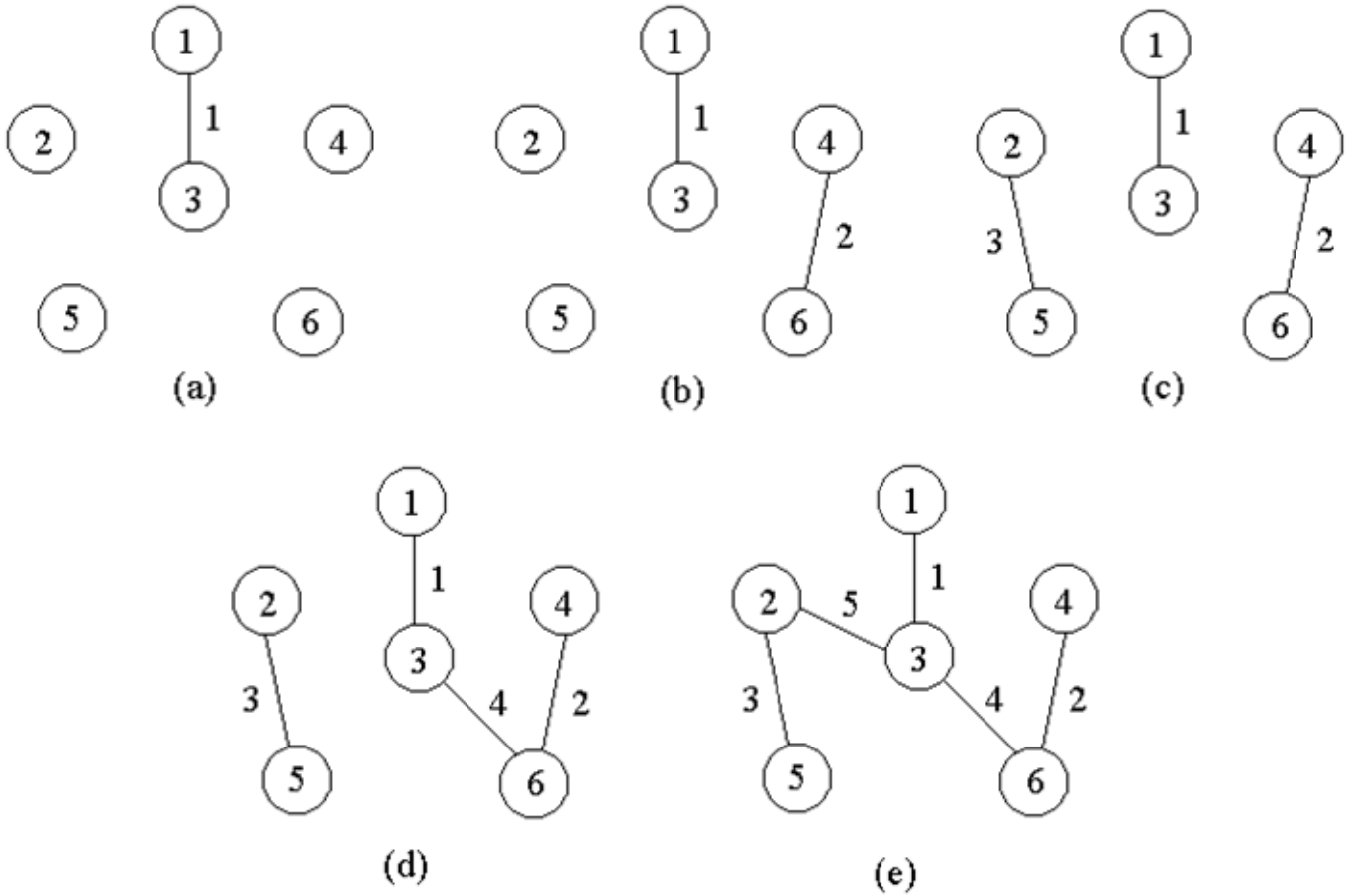
■时间复杂度：$O(n\lg n + nf(n))$

- 排序：$O(n\lg n)$
- 检查$A \cup \{x\}$独立性：$O(f(n))$

定理16.11说明了加权拟阵上贪心算法的正确性

其他应用——最小生成树 (续)





其他应用——最小生成树 (续)

- 当$|E| = \Omega(|V|^2)$时，Kruskal算法比Prim算法差，但当$|E| = O(|V|^2)$时，Kruskal算法却比Prim算法好得多

本章小结

- 贪心算法：启发式思想，每次选择使得当前步骤收益最大化，但并不考虑全局情况，大多数情况无法得到最优解
- 贪心算法可得到最优解：活动选择问题、分数背包问题、Huffman编码、最优装载问题、单源最短路径、最小生成树等
- 寻找加权拟阵的权重最大独立子集
- 贪心算法要素：贪心选择性质、最优子结构

第5章 近似算法

苏州大学 计算机科学与技术学院
汪笑宇
Email: xywang21@suda.edu.cn

本章内容

■近似算法（教材Chapter 35）

- ➢概述
- ➢顶点覆盖问题
- ➢旅行商问题
- ➢集合覆盖问题
- ➢子集和问题

近似算法概述

- 许多具有实际意义的问题都是NPC问题：P≠NP时无法在多项式时间内求最优解
- 解决NPC问题方法：
  - 输入规模小：指数级运行时间算法解决
  - 多项式时间内解决特殊情况
  - 多项式时间内得到近似最优解
- 近似算法：返回近似最优解的算法

近似算法概述 (续)

■最优化问题Π（最小化问题/最大化问题）：

- D：输入实例集合
- $S(I)$：输入实例$I \in D$的可行解集合
- $f : S(I) \to \mathbb{R}$：将每个可行解进行赋值的函数，称为目标函数
- $\mathrm{OPT}(I)$：输入实例I对应最优解的值，即$f(s^*)$的值，其中$s^* \in S(I)$是输入I的最优解
- $\mathrm{SOL}_A(I)$：算法A得到的输入实例I的解的值，即$f(s)$的值，其中$s \in S(I)$是算法A得到的输入I的解

近似算法概述 (续)

■最优化问题Π（最小化问题/最大化问题）：

➢算法A的近似比α：

$$\forall I \in D: \begin{cases} 1 \leq \dfrac{\mathrm{SOL}_A(I)}{\mathrm{OPT}(I)} \leq \alpha, & \text{Minimization} \\ \alpha \leq \dfrac{\mathrm{SOL}_A(I)}{\mathrm{OPT}(I)} \leq 1, & \text{Maximization} \end{cases}$$

近似比限制了算法A所能达到的最坏情况